LEVEL II

1

TR-665

DAAG53-76C-0138

June 1978

DISCRETE RELAXATION FOR MATCHING RELATIONAL STRUCTURES,

Les Kitchen
Computer Science Center
University of Maryland
College Park, MD 20742

COMPUTER SCIENCE TECHNICAL REPORT SERIES

UNIVERSITY OF MARYLAND
COLLEGE PARK, MARYLAND
20742

LEVEL II

(1)

(9) Technical Rept.

(14) TR-665

(15) DAAG53-76C-0138

(11) June 1978

(6) DISCRETE RELAXATION FOR MATCHING RELATIONAL STRUCTURES.

(12) 39 p.

(10) Les Kitchen
Computer Science Center
University of Maryland
College Park, MD 20742

ABSTRACT

Local constraint analysis ("discrete relaxation") is used to reduce ambiguity in matching pairs of relational structures. It is found empirically that if the set of possible local properties is sufficiently large, this generally results in unambiguous identifications after only a few iterations.

(15) The support of the U.S. Army Night Vision Laboratory under Contract DAAG53-76C-0138, ARPA Order-3206 is gratefully acknowledged, as is the help of Mrs. Virginia Kuykendall in preparing this paper.

403 018

Preface

In the past, "relaxation" methods have usually been applied to problems of identifying points in a fixed, simple structure (namely, the pixels in a digital picture array). This report addresses the more general problem of identifying objects which may be parts of an arbitrary structure. Such a situation arises in the higher-level processing of images, where the objects may correspond to regions of an image, extracted by some segmentation process. Regions would be linked by such relations as "is above", "surrounds", or "is larger than".

The first two sections of this report establish some theoretical results concerning the effectiveness of relaxation in such situations. The third section demonstrates the application of the method to some simple, but not unreasonable problems; while the fourth, among other matters, discusses how such relaxation methods could be extended to better handle practical image recognition problems. Some of these extensions will be treated in a future report.

List of Set-Theoretic Symbols not Defined in Text

Use of Symbol	Meaning
$a \in A$	"a is an element of the set A".
$A \subseteq B$	"A is a subset of B" (NB: This does not preclude A=B.)
$A \times B$	The Cartesian product of sets A and B. This is the set of all ordered pairs (a,b) where $a \in A$, $b \in B$.
A^n	The set of all ordered n-tuples of elements of A.
$F: A \rightarrow B$	"f is a function, which takes as argument an element of the set A, and yields an element of B as result".
$\exists$	"there exists".
{a: Property}	The set of all objects, a, for which the mentioned property holds.

ERRATA

Section 3, page 2, line 14: Insert "on the list" after "element".

Section 4, page 2, line 1: Delete quotation mark after A.

1. Relational structures and monomorphisms

Many authors have used finite relational structures as a formalism for describing visual scenes, and for the recognition of known objects in such scenes. (For example, Barrow & Popplestone 1971, Barrow et al. 1972, Winston 1970.) In this section we introduce the notion of a relational structure using a treatment which closely follows Barrow et al. (1972).

Definition 1

A (finite) relational structure is a triple $\langle X, P, \phi \rangle$.

X is a finite set of nodes, called the carrier of the relational structure.

P is a finite set of predicates. Each predicate P has associated with it a positive integer called the order of the predicate. We write $P^{(n)}$ if P has n arguments.

ϕ is a function which maps each predicate $P^{(n)} \in P$ to an n-ary relation on X. That is, $\phi(P^{(n)}) \subseteq X^n$.

For $x_1, x_2, \ldots, x_n \in X$, $P^{(n)} \in P$ we write $P^{(n)}(x_1, x_2, \ldots, x_n)$ precisely when $(x_1, x_2, \ldots, x_n) \in \phi(P^{(n)})$. Conventionally, a relational structure is referred to by the name of its carrier, its two remaining components being left implicit.

Definition 2

Let M and W be two relational structures with the same predicate set P. A one to one mapping $f : M \rightarrow W$ is called a monomorphism of M into W (written $f: M \leq W$) when for all $P^{(n)} \in P$ and $x_1, x_2, \ldots x_n \in M$, $P^{(n)}(x_1, x_2, \ldots, x_n)$ implies $P^{(n)}(f(x_1), \ldots, f(x_n))$. That is, f preserves the structure of M. If there

exists a mapping f : M ≤ W, we will often write merely M ≤ W, the mapping f being left implicit.

The set M can be thought of as a model which describes some object of interest, while W can be thought of as a description of a world, or universe, in which we are searching for an instance of this object. Thus the problem of finding the monomorphisms between two relational structures is of some importance. In general this problem is computationally very difficult. [It is in fact NP-complete, since it is a generalization of the sub-graph matching problem for graphs, a problem which is known to be NP-complete (Read & Corneil 1977).] However, for those relational structures which are commonly encountered, we would hope that the problem is not so intractable. That is, while the worst-case behavior of an algorithm for finding monomorphisms may be exponential, "on the average" it may find solutions quite rapidly.

2. Discrete Relaxation

The term "relaxation" has been used to describe a class of iterative methods for classification or constraint analysis (Davis & Rosenfeld 1977, Rosenfeld 1977, Rosenfeld et al. 1976, Zucker et al. 1976). All of these methods have the following in common: we wish to calculate a value of some sort for each point in a structure. This is done by firstly assigning to each point an initial approximation to this value. Then the approximation at each point is improved (in some sense) by examining the values of the approximations at its neighbors in the structure. This improvement step can be repeated until some condition is satisfied, typically until no further improvement can be made. The "values" referred to above need not be numeric, they may be symbolic labels of some sort, and an "approximation" to such a value might be a list of possible labels, each with some sort of likelihood measure attached. The "structure" referred to is often the grid of pixels which represents a digital picture, and in this case the "neighbors" of a point would be its adjacent pixels on the grid. The above is by no means intended as a definitive description of relaxation methods, but merely to indicate informally the type of processing which all these methods have in common.

The same type of approach can be applied to the problem of matching relational structures, and is developed below. We assume that all relational structures mentioned have the same predicate set P.

Definition 3

An assignment of a relational structure M to a relational structure W (the order being material) is any subset of M×W. Thus an assignment of M to W is a pairing of elements of M with elements of W. It is in a sense a binary relation between M and W, but should not be confused with those relations that are responsible for the internal structures of M and W.

Definition 4

Let M be a relational structure and $x \in M \times W$ the neighborhood of x (written Nbd(x)) is the set:

$$\{y \in M : \exists P^{(n)}, x_1, x_2, \ldots x_n, i, j \text{ such that } P^{(n)}(x_1, x_2, \ldots, x_n) \text{ and } x = x_i \text{ and } y = x_j\}$$

The neighborhood of x consists of all those nodes which are directly related to x in any way by a predicate.

Definition 5

Let R be an assignment of M to W. A pair $(x,y) \in M$ is said to be locally consistent with respect to R when there exists a monomorphism $f: Nbd(x) \leq Nbd(y)$ such that $f(x)=y$, and $f \subseteq R$ considered as a set of ordered pairs.

In other words, R permits a mapping which preserves the local structure around x and y.

Definition 6

An assignment R is said to be locally consistent when every pair in R is locally consistent with respect to R.

The notion of a locally consistent assignment is considerably weaker than that of a monomorphism. The directed graphs of Figure 1 furnish an example: The assignment which pairs every node of the "triangle" with every node of the "square" is locally consistent, but there can be no monomorphism between the two. However, local consistency is a "natural" generalization of monomorphism, as the following theorem shows. Any mapping from M to W, where M and W are relational structures, can be regarded as an assignment of M to W, since a mapping is merely a particular type of pairing. Monomorphisms are precisely those mappings which are injective (i.e., one-to-one) and consistent as assignments.

Theorem 1

Let M and W be relational structures, and let $f : M \rightarrow W$ be injective. The f is a monomorphism if and only if f is locally consistent.

Proof

(Sufficiency)

If f is a monomorphism, then clearly f must be locally consistent.

(Necessity)

Assume f is locally consistent. Let $P^{(n)} \in P$, and $x_1, x_2, \ldots, x_n \in M$, such that $P^{(n)}(x_1, x_2, \ldots, x_n)$. Since f is locally consistent, in particular about x_1, there exists a monomorphism f_1: $Nbd(x_1) \leq Nbd(f_1(x_1))$ where $f_1 \subseteq f$. Thus we have $P^{(n)}(f_1(x_1), f_1(x_2), \ldots, f_1(x_n))$. However, f is a function, so the image

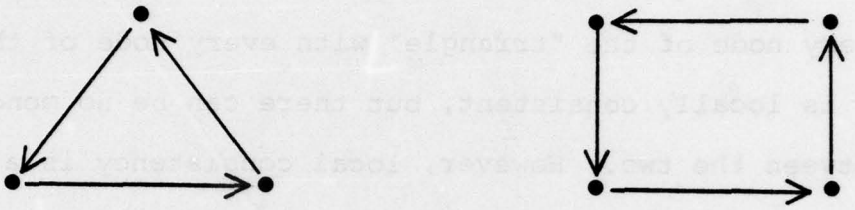

Figure 1. Many locally consistent assignments, but no monomorphism.

of every point in M is uniquely determined. Therefore, $f_1(x_1)=f(x_1)$, $f_1(x_2)=f(x_2)$, etc., and hence $P^{(n)}(f(x_1), f(x_2),\ldots,f(x_n))$. We conclude that f is a monomorphism. QED.

We now display an abstract version of the discrete relaxation algorithm, applied to two relational structures M and W.

$R_0 := M \times W$; $i := 0$;

repeat

$i := i+1$

$R_i := \{$all pairs in R_{i-1} which are locally consistent w.r.t. $R_{i-1}\}$

until $R_i = R_{i-1}$

$R^* := R_i$

This algorithm at each step removes all pairs from the local assignment which are not locally consistent.

It is easy to see how this fits into the relaxation framework. The stucture we are dealing with is the relational structure W. Each assignment can be regarded as attaching a (possibly empty) set of model nodes from M as labels on each world node in W. The improvement step consists of removing those labels which are locally inconsistent. The following results are generalizations of those in Rosenfeld et al. (1976).

Theorem 2

The above algorithm always terminates.

Proof

At the end of each iteration of the algorithm either $R_i = R_{i-1}$ or $R_i \subset R_{i-1}$. If $R_i = R_{i-1}$ the algorithm terminates. Since $R_0 = M \times W$ is a finite set, there can only be a finite number of iterations for which $R_i \subset R_{i-1}$. QED.

Theorem 3

The assignment R* produced by the above algorithm is locally consistent.

Proof

Suppose the algorithm terminates after k iterations, that is $R^* = R_k$. However, R_k = {all pairs in R_{k-1} which are locally consistent w.r.t. R_{k-1}}.

But $R_k = R_{k-1}$, since the algorithm terminated at this step. So all pairs in R_k are locally consistent with respect to R_k, hence $R_k = R^*$ is locally consistent. QED.

Theorem 4

R* is the maximal locally consistent assignment of M to W. That is, if R is any locally consistent assignment of M to W, then $R \subseteq R^*$.

Proof (by induction on the number of iteration steps)

Basis

$R \subseteq R_0 = M \times W$, by definition.

Induction

Suppose $R \subseteq R_{i-1}$ for some $i>0$. Let (x,y) be any pair in

R. This pair is locally consistent with respect to R, and so it is locally consistent with respect to R_{i-1} since $R \subseteq R_{i-1}$. Thus $(x,y) \in R_i$ by the iteration step of the algorithm.

Conclusion

We see that $R \subseteq R_i$ for all i, and in particular $R \subseteq R^*$.

Corollary.

If f is any monomorphism $f: M \leq W$, then $f \subseteq R^*$.

Thus the discrete relaxation algorithm is guaranteed to capture any monomorphisms that exist between two relational structures. In general R* will contain much else besides. Figure 1 again illustrates a pathological case where $R_0 = M \times W$ is locally consistent. However, it seems that in practice the only locally consistent assignments are usually in fact monomorphisms (or trivially the empty assignment). This is borne out by the experiments described below. Even though discrete relaxation may gain nothing at all, it often will in practice provide solutions to the problem of finding monomorphisms between relational structures. The reader should note that if there are several distinct monomorphisms between two relational structures M and W, then at best R* will contain the union of all these monomorphisms and there still remains the problem of disentangling the individual monomorphisms.

The basic technique described above is not new. Calling it "refinement", Ullmann (1976) used it for the more restricted problem of subgraph matching. Under the name "reduction" it has been used by Haralick (1977, et al. 1978) in a more general

"Theory of Arrangements". Ullmann used his refinement as an adjunct to a conventional tree-search procedure for detecting subgraph isomorphism. The same technique could be used here, and would be most useful for those cases where discrete relaxation makes considerable, but only partial, progress towards finding monomorphisms.

Note that in order to determine whether a given pair (x,y) is locally consistent with respect to an assignment R_{i-1}, we must still conduct a search for a monomorphism $f:Nbd(x) \leq Nbd(y)$, with the other required properties, $f \subseteq R_{i-1}$ and $f(x)=y$, and this search must be repeated for every pair in R_{i-1}. Compared to a tree-search procedure for detecting monomorphisms, we have traded one large combinatorial search for many small combinatorial searches. This would seem to make discrete relaxation considerably faster. Furthermore, the local consistency checks for all the pairs in R_{i-1} are independent, and could therefore be carried out in parallel, if suitable hardware were available. The price we pay for these advantages is the loss of any guarantee of finding the monomorphisms between two structures.

It is possible to use a weaker characterization of consistency which obviates the search for a monomorphism between neighborhoods.

Definition 7

Let M and W be relational structures, and let R be an assignment of M to W. A pair $(x,y) \in M \times W$ is said to be weakly consistent with respect to R when there exists an assignment Q of Nbd(x)

to Nbd(y) with the following properties:

(i) $Q \subseteq R$;

(ii) $(x,y) \in Q$;

(iii) $(x,z) \in Q$ implies $y = z$;

(iv) for all $P^{(n)}$, and for all $x_1, x_2, \ldots, x_n \in M$, $x \in \{x_1, x_2, \ldots, x_n\}$ implies that there exist $y_1, y_2, \ldots, y_n \in W$ such that $(x_1,y_1), (x_2,y_2), \ldots, (x_n,y_n) \in Q$ and $P^{(n)}(y_1, y_2, \ldots, y_n)$. That is, for every predicate instance in the model which mentions x, R permits a renaming of arguments so that the predicate holds in the world. This renaming need not be one to one, nor even a function, but x must always be replaced by y.

Weak consistency can be used instead of local consistency in a discrete relaxation process, and corresponding theorems can be proved. However, the results it produces are more likely to be ambiguous, in that weak consistency will permit pairings which are locally inconsistent. It should be noted that weak consistency corresponds to the notion of consistency introduced by Rosenfeld et al. (1976).

3. Experiments

A computer program has been written which implements the discrete relaxation algorithm described above. There are, however, several minor differences. The initial iteration is anomalous. Since $R_0 = M \times W$, we can check the local consistency of pairs in R_0 "on the fly" by examining all pairs systematically, without the need to store them. Furthermore, since on this iteration many pairs are examined, the program uses the simpler weak consistency condition. This does not invalidate any of the results proved in the last section, but is easier to test. All subsequent iterations use the full local consistency condition. Obviously, the whole sequence of assignments $R_1, R_2, \ldots, R^*$ need not be stored; only the current version is . On each iteration those pairs in the current assignment which are not locally consistent are flagged, and at the end of each iteration these flagged pairs are deleted from the assignment. This means that the algorithm terminates whenever no deletions are made on a given iteration.

The remaining difference is in the testing of the local consistency of a pair (x,y). Rather than search for an appropriate monomorphism of Nbd(x) into Nbd(y), the program searches for an appropriate partial function from Nbd(y) to Nbd(x) which preserves structure in a reverse sense. The two types of maps are essentially equivalent (in that the existence of a map of the one sort implies the existence of a map of the other), but the second characterization was more convenient to use in the context of the program's data structures.

The data and instructions of the program were organized to avoid any gross inefficiences; however, no attempt was made to optimize its performance. The program itself, about 900 lines long, is written in Pascal, and runs on the Univac 1108.

At first the program was tried in an ad hoc fashion on the matching of various relational structures, some concocted by hand, others based on the adjacencies of states in political maps. On these the program generally performed quite well, and encouraged by these results we conducted the more systematic tests described below.

The program was modified to generate random sub-structures of a relational structure read as input. The method used was this: An element of the structure is chosen at random, and used to start a list. Then repeatedly an element which has neighbors not on the list is chosen at random, next one of its neighbors not on the list is chosen at random and placed on the list. This continues until the list reaches some pre-specified size. The list is then used as the carrier of a relational structure which inherits from the original relational structure all those predicates which have as arguments only elements which appear on the list. This process tends to generate compact substructures, embedded in a larger relational structure, and thus corresponds to the intuitive notion of compact objects embedded in a large visual scene. The program then attempts to match the substructure generated as model with the larger parent structure as world. In this case a monomorphism certainly exists, so we can measure how often the program finds it, and how much computation the relaxation takes.

The data for the first group of tests was derived from the political map of Africa, which has 46 countries. The map is described in terms of the adjacencies of the countries, together with some of the following properties:

a) Whether a country is coastal or inland

b) The number of letters in its name

c) The first letter of its name

d) Its color on the globe (National Geographic Society, 1976)

e) The first letter of the name of its capital city

We thus have a relational structure with one binary predicate, adjacent, and a large set of possible unary predicates.

Tests were made using the adjacency data together with various subsets of the properties. All tests were run on the same randomly generated models of sizes ranging from 4 to 13 nodes, using five examples of each size.

The results of these tests are summarized in Tables 1-5. In each of these tables, the column headed "Iterations" gives the average number of iteration steps required for convergence. The column headed "Comparisons" gives the average number of times (000 omitted) that it was necessary to compare one predicate with another in the matching; thus it is an indication of the computational effort required by the relaxation process. The column headed "Ambiguity" measures the average number of excess pairs remaining after convergence is complete; it is zero if there is exactly one pair for every element in the model. All of these averages are taken over the five instances

Model Size	Iterations	Comparisons	Ambiguity
4	3.4	4.2	0
5	3.6	7.0	0.6
6	4.0	8.6	0.2
7	3.8	11.1	1.2
8	4.0	11.0	1.0
9	3.8	14.4	0.4
10	4.0	17.2	1.2
11	4.2	18.1	2.0
12	4.0	20.8	0.8
13	4.2	23.3	0.4

Table 1. First letter of name (Africa)

Model Size	Iterations	Comparisons	Ambiguity
4	3.4	5.2	0
5	3.4	8.1	0
6	3.4	10.3	0
7	3.4	12.7	0.4
8	3.6	13.4	0.6
9	3.4	16.9	0.2
10	3.6	18.7	0.8
11	3.8	21.4	1.2
12	3.8	23.2	0.2
13	3.4	25.8	0

Table 2. First letter of name; coastal/inland (Africa)

Model Size	Iterations	Comparisons	Ambiguity
4	2.6	4.9	0.0
5	3.0	7.6	0.0
6	2.8	9.8	0.0
7	3.0	12.0	0.2
8	3.0	12.5	0.0
9	3.0	16.1	0.2
10	3.0	17.5	0.4
11	3.0	19.8	0.6
12	3.0	21.7	0.6
13	3.2	24.8	0.2

Table 3. First letter of name; color on globe (Africa)

Model Size	Iterations	Comparisons	Ambiguity
4	3.2	4.3	1.4
5	3.6	6.7	1.0
6	3.6	8.3	0.2
7	3.8	10.4	0.6
8	3.6	11.8	1.2
9	4.0	16.2	1.2
10	3.8	16.3	1.8
11	3.8	20.3	1.6
12	4.0	21.4	1.2
13	3.8	22.3	0.8

Table 4. First letter of name of capital city (Africa)

Model Size	Iterations	Comparisons	Ambiguity
4	2.4	4.9	0
5	2.6	7.6	0.2
6	2.8	9.8	0
7	3.6	12.4	0.2
8	3.4	12.8	0.2
9	3.4	16.6	0.2
10	3.2	17.8	0.4
11	3.2	20.1	0.0
12	3.4	22.5	0.0
13	3.4	25.2	0.0

Table 5. First letter of name of capital city; length of name (of country) (Africa)

of each model size. [The predicates used are indicated in the table captions; the adjacency relation was also used in every test.]

A second, similar group of tests was run, using corresponding data derived from the 48 contiguous states of the U.S.A. The results of these tests are shown in Tables 6-10.

It is seen from these tables that if there is a reasonable variety of different predicates, so that local inconsistencies are easily found, the results are quite unambiguous, and convergence is rapid (3 to 4 iterations). The remaining ambiguity was most often due to the existence of multiple monomorphisms for the chosen models. It is interesting to note that the number of iterations does not seem to depend significantly on the model size. The number of comparisons does increase with the model size, but only at an essentially linear rate.

A version of the program which uses weak consistency for all iterations was also tried. The results of using this version on the same data as used in Table 1 are shown in Table 11. Note that the number of iterations required has increased slightly, but the number of comparisons has decreased, while the ambiguity remains the same.

Model Size	Iterations	Comparisons	Ambiguity
4	4.0	4.9	0.2
5	3.4	5.8	1.0
6	4.0	7.9	0.2
7	4.2	9.3	0.2
8	3.6	12.1	0.6
9	4.0	14.0	1.4
10	3.8	14.4	1.0
11	4.0	17.6	0.6
12	3.6	18.5	0.4
13	4.0	22.1	0.2

Table 6. First letter of name (U.S.A.)

Model Size	Iterations	Comparisons	Ambiguity
4	3.8	6.0	0.0
5	3.2	7.0	1.0
6	3.6	9.5	0.2
7	3.8	11.2	0.2
8	3.6	14.7	0.4
9	3.8	16.8	0.8
10	3.6	17.3	1.0
11	3.6	21.0	0.6
12	3.6	22.6	0.4
13	3.6	26.6	0.2

Table 7. First letter of name; coastal/inland (U.S.A.)

Model Size	Iterations	Comparisons	Ambiguity
4	2.8	5.7	0.2
5	3.2	6.8	0.0
6	3.0	9.2	0.0
7	3.2	10.6	0.0
8	3.2	14.3	0.2
9	3.0	15.9	0.0
10	3.0	16.8	0.2
11	3.0	19.8	0.2
12	3.2	22.2	0.0
13	3.0	24.9	0.0

Table 8. First letter of name; color on globe (U.S.A.)

Model Size	Iterations	Comparisons	Ambiguity
4	3.4	4.8	0.4
5	3.6	5.8	1.0
6	3.6	7.7	0.2
7	3.6	8.6	0.4
8	3.6	12.1	0.6
9	3.8	13.3	0.0
10	4.2	14.3	0.4
11	3.8	16.8	0.8
12	4,0	19.1	0.6
13	4.0	21.8	0.2

Table 9. First letter of name of capital city (U.S.A.)

Model Size	Iterations	Comparisons	Ambiguity
4	2.8	5.7	0.0
5	2.8	6.7	0.0
6	3.0	9.2	0.0
7	3.0	10.5	0.0
8	3.0	14.2	0.0
9	3.0	15.8	0.0
10	3.0	16.8	0.0
11	3.0	19.7	0.0
12	3.0	22.0	0.0
13	3.0	24.8	0.0

Table 10. First letter of name of capital city; length of name (of state) (U.S.A.)

Model Size	Iterations	Comparisons	Ambiguity
4	3.4	4.1	0.0
5	3.8	6.4	0.6
6	4.0	8.2	0.2
7	4.0	10.2	1.2
8	4.0	10.4	1.0
9	4.2	13.5	0.4
10	4.4	14.9	1.2
11	4.4	16.8	2.0
12	4.6	18.6	0.8
13	4.4	20.8	0.4

Table 11. First letter of name (weak consistency--Africa)

4. Discussion and extensions

We have described an algorithm which provides an "approximate" solution to the problem of finding monomorphisms between relational structures. Instead of finding monomorphisms, the discrete relaxation algorithm produces a set of ordered pairs which is guaranteed to contain all the monomorphisms between two structures, but this set may contain much else besides. However, in reasonable, practical cases, relaxation finds exact solutions, or very nearly exact solutions, and finds them quickly. This is not to say that a conventional tree-search method would not perform as well, for the very circumstances under which relaxation works well would permit considerable pruning of a search tree. We wish merely to remark that relaxation appears to be a useful tool for solving such problems, either alone, or in conjunction with a tree-searching procedure.

To this end, some improvements could be made in the present program. First of all, some attention should be given to the order in which predicates are checked when testing for local consistency. If rarer predicates are examined first, it is more likely that a locally inconsistency will be discovered early. Secondly, since unary predicates can affect local consistency only on the first iteration of relaxation, they should be disregarded on all subsequent iterations. Thirdly, provision should be made for special treatment of commonly occurring types of relations. For example, each instance of a symmetric binary relation (like adjacency in the previous

section) must be stored twice, once in each sense ("A" is adjacent to B" and "B is adjacent to A"). Special handling of symmetric relations could reduce space and processing requirements considerably.

Beyond this, extensions can be made both to the notion of relational structures, and to the application of relaxation techniques. A first step is the introduction of quantitative predicates. A quantitative binary predicate Distance would indicate not only that its two arguments have a distance between them, but would provide the numeric value of this distance, for instance Distance (Earth, Moon, 400,000km). Two quantitative predicates would agree when their numeric values were equal, or nearly equal relative to some error tolerance. Such predicates pose few problems in themselves, since each quantitative predicate could be replaced by a whole family of ordinary qualitative predicates, one for each value that the numeric quantity can take according to some more or less fine quantization.

However, quantitative predicates soon lead us to consideration of fuzzy or probabilistic truth. The more closely two quantitative predicates agree, the more likely it is that they actually match. Further, the real-world data we use may be subject to measurement errors with possibly known distributions. Even qualitative information may be more or less certain. Relaxation can be readily adapted to handle such concepts by associating with each predicate some sort of truth measure,

and associating with each pair in an assignment some measure of its local consistency. The relaxation iteration would adjust this local consistency measure based on the local consistency values of its neighbors, and on the truth values of predicates that link it with these neighbors. Of course many details remain to be filled in, but the basic approach is clear.

In this application, a relaxation-type technique has the advantage that, since all local consistency checks are independent, the result produced is not affected by the particular order in which the checks are performed. Thus the results produced by relaxation would tend to be more stable and reliable than those produced by tree-search methods.

Another topic worth pursuing is the development of more general models. While a relational structure can adequately represent a particular view of a particular object, it cannot easily describe the three-dimensional structure of an object (at least in a manner which allows different views of it to be recognized). Nor can it describe a whole class of objects, generally similar, but individually distinct, in a way which permits particular instances of that class to be recognized. Obviously, such problems are very difficult; however, it may be possible to make some progress towards solving them. For example, a relational structure is implicitly a conjunction of all the predicate instances in it. We could use as models structures which permit other logical connectives. It may

be possible to match such models against relational structures without invoking the full generality of predicate calculus and mechanical theorem proving. It would also be useful to have models that are structured in a hierarchical fashion from smaller pieces. This permits economy of description, since commonly occurring pieces need not be duplicated for every instance. It would also simplify the modelling of complex objects.

Relational structures provide an adequate mechanism for describing visual images, since an image is a static sort of thing: a particular set of objects, with particular properties, all in particular relations with each other. However, such a use of relational structures can only be effective if the image in question has been property segmented to begin with. The regions, edges and other features extracted from the image must correspond closely with real objects; otherwise the derived relational structure will fail to be an accurate representation ofthe scene. With this in mind, it may be desirable to create the relational structure which describes the image "on the fly" so to speak, during the matching process. That is, rather than segment the image once and for all, and then build a relational structure from the pieces, we could use the matching process to guide the segmentation, extracting features and properties from the image only as needed to match part of the model.

Of course, much more could be written on the subject of recognizing objects in pictures. Suffice to say that

relational structures (or their extensions) provide a convenient data structure for describing objects and pictures, and relaxation is a useful method for operating on them.

REFERENCES

1. Barrow, H.G. & Popplestone, R.J. (1971) "Relational descriptions in picture processing", in Machine Intelligence 6, B. Meltzer & D. Michie, eds., American Elsevier, New York, pp. 377-396.

2. Barrow, H.G., Ambler, A.P. & Burstall, A.M. (1972) "Some techniques for recognizing structures in pictures", in Frontiers of Pattern Recognition, S. Watanabe, ed., Academic Press, New York, pp. 1-29.

3. Davis, L.S. & Rosenfeld, A. (1977) "Hierarchical relaxation for waveform parsing", Technical Report 568, Computer Science Center, University of Maryland, College Park, MD.

4. Haralick, R.M. (1977) "Scene analysis, arrangements and homomorphisms", Technical Report 527, Computer Science Center, University of Maryland, College Park, MD.

5. Haralick, R.M., Davis, L.S., Rosenfeld, A. & Milgram, D.L. (1978) "Reduction operations for constraint satisfaction", Information Sciences, Vol. 14, pp. 199-219.

6. Read, R.C. & Corneil, D.G. (1977) "The graph isomorphism disease", Journal of Graph Theory, Vol. 1, pp. 339-363.

7. Rosenfeld, A. (1977) "Iterative methods in image analysis", IEEE Conf. on Pattern Recognition and Image Processing, pp. 14-18.

8. Rosenfeld, A., Hummel, R. & Zucker, S. (1976) "Scene labeling by relaxation operations", IEEE Trans. Systems, Man and Cybernetics, Vol. 6, pp. 420-233.

9. Ullman, J.R. (1976) "An algorithm for subgraph isomorphism", Journ. Assoc. Computing Machinery, Vol. 23, pp. 31-42.

10. Winston, P. H. (1970) "Learning structural descriptions from examples", Project MAC Technical Report 76, Massachusetts Institute of Technology, Cambridge, Massachusetts.

11. Zucker, S.W., Krishnamurthy, E.V., & Haar, R.L. (1978) "Relaxation processes for scene labeling: convergence, speed and stability", IEEE Trans. Systems, Man and Cybernetics, Vol. 8, pp. 41-48.

12. National Geographic Political Globe, National Geographic Society, Washington, D.C., 1976.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* Discrete Relaxation for Matching Relational Structures		5. TYPE OF REPORT & PERIOD COVERED 6. PERFORMING ORG. REPORT NUMBER TR-665
7. AUTHOR(s) Les Kitchen		8. CONTRACT OR GRANT NUMBER(s) DAAG53-76C-0138
9. PERFORMING ORGANIZATION NAME AND ADDRESS Computer Science Center University of Maryland College Park, MD 20742		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS U.S. Army Night Vision Lab. Ft. Belvoir, VA 22060		12. REPORT DATE June 1978 13. NUMBER OF PAGES 34
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)*		15. SECURITY CLASS. *(of this report)* Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Subgraph matching
Relaxation
Constraint analysis

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

Local constraint analysis ("discrete relaxation") is used to reduce ambiguity in matching pairs of relational structures. It is found empirically that if the set of possible local properties is sufficiently large, this generally results in unambiguous identifications after only a few iterations.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*